LSDT API
DYNAMIC LINK LIBRARY
REFERENCE GUIDE

LUMISYS

LSDT API Dynamic Link Library

Reference Guide

P/N 0068-105, Rev 08

March 9, 1999

Lumisys, Inc.
225 Humboldt Court
Sunnyvale, CA 94089

TABLE OF CONTENTS

TABLE OF CONTENTS **1**

1. INTRODUCTION **4**
1.1 Document Notation 4
1.2 Terminology 4
1.3 Related Documents 4
1.4 Environment Requirements 4
1.4.1 Minimum Hardware 4
1.4.2 Operating Systems 4
1.4.3 CPU Resources Used by the Data Control Board 5
1.4.4 LSDT Device Driver (DCB Interface Only) 5

2. LUMISCAN LSDT ARCHITECTURE **6**
2.1 Interfaces 6
2.1.1 DCB Interface 6
2.1.1.1 DCB Memory Mapping 6
2.1.2 SCSI Interface 7
2.2 Technical Features 7
2.2.1 Scan Mode and Pixel Resolution 7
2.2.1.1 Fixed 7
2.2.1.2 Variable 7
2.2.2 Pixel Depth 7
2.2.3 Pixel Format 8
2.2.4 Averaging Modes 8
2.2.5 Shifted Dynamic Range Mode 8
2.2.6 User LUT 8
2.2.7 6-Sheet Film Feeder 9
2.2.8 Bulk Autoloader Film Feeder 10
2.2.9 Barcode Reader 10
2.2.10 Film Present Sensor 10
2.3 Digitizer Capabilities 12

3. SOFTWARE FUNCTIONS **13**
3.1 General Setup Functions 13
3.2 Special Operation Functions 13
3.3 Scanning Functions 13
3.4 Post-Scan Functions 13
3.5 Miscellaneous Functions 14
3.6 Diagnostic Functions 14
3.7 Compatibility Functions 14

4. SOFTWARE APPLICATIONS **15**
4.1 Linking LSDT Software Functions 15
4.2 Sample Code 15
4.2.1 Very Simple Example 15
4.2.2 Better Example 16
4.2.3 Finding Pixels Per Line 18

5. LIBRARY FUNCTIONS REFERENCE **19**
5.1 LS_ChangeTimer 19
5.2 LS_CloseDevice 19
5.3 LS_DiagnosticScan 20
5.4 LS_DriverStateString 20
5.5 LS_ErrorString 21
5.6 LS_FeedFilm 21

5.7 LS_FileCopy 22
5.8 LS_GetAllTimers 22
5.9 LS_GetApiInfo 23
5.10 LS_GetDataCount 23
5.11 LS_GetDriverHandle 23
5.12 LS_GetDrvInfo 24
5.13 LS_GetDrvVersion 25
5.14 LS_GetImageWindow 25
5.15 LS_GetLastSCSIError 25
5.16 LS_GetPixelsPerLine 25
5.17 LS_GetStatus 26
5.18 LS_GetTimer 26
5.19 LS_GetVersion 27
5.20 LS_isBarCodeValid 27
5.21 LS_isFilmPresent 27
5.22 LS_LoadCalibrationTable 28
5.23 LS_LoadCorrectionLookUpTable 28
5.24 LS_LoadLookUpTable 28
5.25 LS_LoadLut 29
5.26 LS_MapImageData 30
5.27 LS_MapImageWindow 31
5.28 LS_ModelNameString 31
5.29 LS_Motor 32
5.30 LS_OpenDevice 32
5.31 LS_PixelDepthString 32
5.32 LS_PixelFormatString 33
5.33 LS_ReadBarCode 33
5.34 LS_ReadCalibrationTable 34
5.35 LS_ReadCorrectionLookUpTable 34
5.36 LS_ReadDiagnostic 34
5.37 LS_ReadImageIntoBuffer 35
5.38 LS_ReadImageIntoFile 36
5.39 LS_ReadImageIntoFile_Fast 36
5.40 LS_ReadImageToPointer 37
5.41 LS_ReadLookUpTable 37
5.42 LS_ReadLut 38
5.43 LS_RegisterCmd 39
5.44 LS_ResetDriverVariables 39
5.45 LS_ResetHardware 39
5.46 LS_ResetScan 40
5.47 LS_ScanModeString 40
5.48 LS_SetAveragingMode 40
5.49 LS_SetDevNum 41
5.50 LS_SetLaserMode 41
5.51 LS_SetNextScanParams 41
5.52 LS_SetPMTvoltage 42
5.53 LS_SetSDRMode 42
5.54 LS_SetTrim 43
5.55 LS_StartFilmScan 43
5.56 LS_StartFilmScanNoCLUT 44
5.57 LS_StartMotor 45
5.58 LS_StatusOfLastScanString 45
5.59 LS_StopMotor 45
5.60 LS_StopScanEject 46
5.61 LS_UnMapImageWindow 46
5.62 LS_WriteMemory 46

APPENDIX A: LUMISYS IMAGE HEADER FORMAT 47

APPENDIX B: FUNCTION ERROR CODES 48

APPENDIX C: SCSI ERROR CODES 49

1. INTRODUCTION

This document describes the functions in the LSDT Application Programmer's Interface Dynamic Link Library (LSDTAPI.DLL). They provide a ***Windows 3.x / Windows 9x / Windows NT*** high-level application interface to LSDT digitizers. Refer to the LSDT Software Functions Library Reference Guide (P/N 0066-022) for information on ***DOS*** high-level application interface functions to the LSDT digitizers.

1.1 Document Notation

ALL_CAPS_IN_ITALICS – denotes a #DEFINE value. Almost all of the #DEFINEs you will need are in the following header files:

- LS_TYPES.H
  Data types and defines used by the library.
- LS_DT.H
  Defines, constants and structures used by the library, LSDT errors, scan modes, LSDT states (a.k.a. "phases"), and miscellaneous defines and structures.
- LS_API.H
  API function declarations for Windows. They are the same for both Win16 and Win32.

If you don't find what you need in these two files, then look in one of the other include files in \INCLUDE (Win16) or \INC (Win32).

1.2 Terminology

- DACQ Data Acquisition Board in the LSDT digitizer
- DCB ISA-based Data Control Board located in the PC
- Host PC containing the DCB
- Win16 for Microsoft Windows 3.1, Windows For Workgroups 3.11, Windows 95, or Windows 98
- Win32 for Microsoft Windows 95, Windows 98, or Windows NT

1.3 Related Documents

LSDT Software Functions Library Reference Guide	P/N 0066-022
LSDT API Dynamic Link Library Reference Guide	P/N 0068-105
LUMISCAN 20 Operator's Reference Guide	P/N 0069-384
LUMISCAN 50 / 75 / 85 Operator's Reference Guide	P/N 0061-494
ACR-2000 Operator's Reference Guide	P/N 0070-711
LUMISCAN LSDT Film Digitizer / CR Reader Configuration Guide	P/N 0071-434

1.4 Environment Requirements

1.4.1 Minimum Hardware

- 486DX2/66 MHz (or faster, depending upon Windows Operating System)
- 8 MB RAM (or more, depending upon Windows Operating System)

1.4.2 Operating Systems

- Windows 3.1 / Windows For Workgroups 3.11 (using 16-bit driver w/ DCB)
- Windows 95 / Windows 98 (using 16-bit driver w/ DCB or 32-bit w/ SCSI)
- Windows NT (using 32-bit driver w/ DCB or 32-bit w/ SCSI)

1.4.3 CPU Resources Used by the Data Control Board

- Address Space
  - 32-Kbyte (0x8000 Hex) window
  - starting address 0xA0000-0xE8000 (switch selectable)
  - 0xD0000 – factory setting
- IRQ
  - 3, 4, 5, 6, 7 (jumper selectable)
  - 5 – factory setting
- I/O Address
  - 0x100-0x11F, 0x120-0x13F, 0x140-0x15F or 0x160-0x17F (switch selectable)
  - 0x100 – factory setting
  - **NOTE**: Older DCB models only supported addresses beginning at 0x100 and 0x120.
- DMA
  - None
- Bus Type
  - 8- or 16-bit ISA

1.4.4 LSDT Device Driver (DCB Interface Only)

For all Win16 systems, the DOS LSDT device driver "LSDTVxxx.COM" must be installed on the host. At the command line or in AUTOEXEC.BAT, enter "LSDTVxxx" (where xxx specifies version number). This must be installed prior to starting Windows. For Windows 95 / Windows 98, the driver must be loaded by AUTOEXEC.BAT. The driver switch options are described in the **LUMISCAN** Operator's Reference Guides.

For Windows NT, the LSDT device driver is automatically loaded. It may be manually loaded by using "Devices" (in "Control Panel") or by typing "NET START LSDT" from a command prompt. See README.TXT for installation instructions for the Windows NT LSDT device driver, LSDT.SYS.

Digitizers with a SCSI interface do not use the LSDT device driver.

2. LUMISCAN LSDT ARCHITECTURE

The LumiScan DeskTop (LSDT) family consists of film digitizers and CR phosphor plate readers using either a scanning laser beam and light collection cylinder or a CCD array with a proprietary illumination system. A pinch roller system is used to transport the medium being scanned.

NOTE: In this document, when we use the term "film", we mean a somewhat generic reference to "whatever is being scanned". Currently, several types of media are used in various digitizers and readers:

• Film (photographic negatives)	LS20 LS50, LS75, LS85	CCD Digitizer Laser Digitizers
• Phosphor-coated CR plates	ACR-2000	CR Reader
• Fluorescent CR Test patterns	ACR-2000	CR Reader

NOTE: Additionally, in this document, we use the term "digitizer" to mean a somewhat generic reference to either a (film) digitizer or a (CR) reader.

An LSDT digitizer houses a Data Acquisition Board (DACQ) which is connected to a Data Control Board (DCB) located in a host computer via a shielded 37-pin cable (for the SCSI interface, the DCB is located inside the digitizer). All operations are controlled by the DCB. Once a scan is initiated, the data acquisition is automatic, requiring no intervention. During a scan operation image data is written into a circular buffer in the image memory on the DCB. On digitizers with enough memory to hold a complete image it is possible to let the scan complete before transfer image data out of DCB image memory. However, if the image size is greater than the circular buffer size, then the situation is entirely different:

NOTE: For Images Greater Than the DCB Memory Size

The pinch roller media transport system does NOT allow the scan operation to stop before the end of the media. Therefore, the host application **MUST** read the image data out of the circular buffer as the data is acquired or the buffer will eventually overflow. When the DCB image memory circular buffer is full it wraps and begins overwriting the oldest data. If the host application has not read the old data when the image memory wraps, a "***DATA_LOST***" error is generated and data from the scan will truly be lost.

2.1 Interfaces

2.1.1 DCB Interface

In this interface, the DCB is an ISA card in the host computer. It connects to the digitizer via a shielded 37-pin cable. The card itself is a PC/AT ISA 8-bit board requiring a 32K byte memory window (0xA0000-0xE8000) and 32 bytes of I/O space based at either 0x100, 0x120, 0x140 or 0x160 (older DCB boards only supported address at 0x100 and 0x120). An interrupt resource (IRQ) is also used and can be set to 3, 4, 5, 6 or 7. The 32K memory window is for access to the image memory and can be placed on any 32K boundary within the 0xA0000-0xE8000 address space. The DCB is populated with 4, 12 or 16 Megabytes of memory depending on the model of digitizer that will be connected.

The DCB is controlled by the LSDT device driver running on the host computer. The LSDT Application Programmer's Interface Dynamic Link Library (LSDTAPI.DLL) provides a run-time Windows application interface to the LSDT digitizer (via the LSDT device driver or SCSI interface). The library contains all the functions needed by a Windows application program to control the LSDT family of digitizers. DOS programs use the LSDT Software Functions Library (LSDTFCNS.LIB) which provide substantially the same functions.

Neither direct interface to the DCB nor direct interface to the DOS/Windows or Windows NT drivers are supported by Lumisys.

2.1.1.1 DCB Memory Mapping

The 32KB address space can be mapped into the application's address space. This allows direct access to the image data. The function LS_MapImageWindow is used to add the DCB memory to the applications address space; LS_UnMapImageWindow is used to remove it. The function LS_GetImageWindow returns the physical address of the DCB memory, which is called by the LS_MapImageWindow function. The function LS_MapImageData causes

the driver to program the DCB hardware to place the image data specified by byte offset from the beginning of the image, into the DCB memory window. All mapping must be on 32K boundaries.

2.1.2 SCSI Interface

In this interface, the DCB is located inside the digitizer. It is part of a single board computer, also inside the digitizer, which in turn connects to a host computer via a standard SCSI cable.

2.2 Technical Features

When you start a scan (LS_StartFilmScan), there are four things you must specify:

- Scan Resolution Mode — Fixed or Variable number of pixels per inch
- Pixel Resolution — Pixels Per Line (PPL) or Pixels Per Inch (PPI)
- Pixel Depth — 8 or 12 bits per pixel
- Pixel Format — LSB or MSB first

Additionally, there are other settable features such as averaging, shifted dynamic range, lookup tables, etc.

2.2.1 Scan Mode and Pixel Resolution

There are two scan modes: FIXED_RESOLUTION and VARIABLE_RESOLUTION. You specify which when you start a film scan. Note that all digitizers have minimum and maximum “pixels per line” and “pixels per inch” resolutions. These minimums and maximums vary according to the digitizer type, and in many cases, to the width of the film being scanned.

2.2.1.1 Fixed

FIXED_RESOLUTION means that the pixel resolution (or “pixels per inch”) will be the same regardless of film width. The resolution will be determined by the “pixels per inch” argument passed to the LS_xxx functions. The number of pixels per line returned is determined by the film width with a maximum being imposed by the digitizer model.

When the requested pixels per inch times the film width causes the total number of pixels per line to exceed the digitizer model hardware limits, an error (***PIXELS_PER_LINE_OUT_OF_RANGE***) is returned and the film is reversed out of the digitizer. However, if the film width is just a little too large (< 1 inch), then the scan is performed by trimming the edges (instead of returning an error).

2.2.1.2 Variable

VARIABLE_RESOLUTION means that the pixel resolution (or “pixels per inch”) will vary according to the width of the film. The resolution will be determined by the “pixels per line” (argument passed to the LS_xxx functions) over the entire width of the film. So regardless of film width, each scanned line will contain PPL pixels. For example, if the pixels per line is set to 1024 and the scan mode is variable, then a scanned line of film will contain 1024 pixels. If the width of film is 14", the pixel resolution will be 1024 ÷ 14 = 73 pixels/inch. If the film width is only 8 inches, a scanned line of film will still contain 1024 pixels; however, the pixel resolution will be higher: 1024 ÷ 8 = 128 pixels/inch.

NOTE: If the film is too narrow, a minimum pixel resolution may be encountered depending on the PPL requested for the scan. The effect of this limit is that the PPL scanned may be less than the PPL requested. The application should **always** check to ensure that the requested PPL are equal to the actual PPL. When should you check? Check right after data first starts becoming available from the scan. If fewer pixels per line are being delivered, the application should adjust accordingly.

In the case of the LS20 (since it is a CCD digitizer), if it scans a film which is too narrow it will adjust simply by scanning “air” on each side of the film.

2.2.2 Pixel Depth

The LSDT digitizer family can scan either 8-bit pixels or 12-bit pixels. In 8-bit mode, the scanning process is the same as 12-bit mode. The hardware digitizes raw 12-bit values and applies the corrections and calibrations to the 12-bit data. The 12-bit data is passed through the User LUT but only the low order 8 bits are placed into the

DCB memory. To get good 8-bit data, the User LUT must map the raw 12-bit data down to the low 8 bits. See the section on the User LUT for details.

Some LSDT library functions require a "pixel depth" parameter. "pixel depth" can have one of two values: SCAN_8BITS or SCAN_12BITS. If “pixel depth” = SCAN_8BITS, then a scanned image results in 1 byte per pixel. If “pixel depth” = SCAN_12BITS then a scanned image results in 2 bytes per pixel. The data format for 12-bit pixels is with all the data in the lower bits.

NOTE: When scanning 8-bit pixels, a special User LUT must be loaded.

2.2.3 Pixel Format

There are two ways that 12-bit data is sent from a digitizer to the host. In both cases, the 12 bits of data are sent as a right-justified value in a 16-bit word. This 16-bit word is sent as Least Significant Byte (LSB) first or Most Significant Byte (MSB) first. You specify which when you start a film scan (LSB_FIRST or MSB_FIRST).

2.2.4 Averaging Modes

There are three averaging modes: none, X only, and X+Y. For X averaging, two raw pixels are input and one pixel (the average) is output. For example, to scan with 1024 pixels per line with X averaging enabled, the hardware would scan 2048 raw pixels per line. Currently there is an absolute limit of 5120 raw pixels per line for all digitizer models. This limits all digitizers to a maximum of 2560 pixels per line with X averaging enabled. Not all digitizer models are capable of producing “absolute limit” pixels per line. Not all digitizer models support all averaging modes. For X+Y averaging, in addition to the X averaging described above, two raw lines are input and one line is output.

In general, scanning with averaging modes enabled improves the signal to noise ratio at the expense of a slower scanning speed.

NOTE: The CCD digitizers (LS20) use averaging to produce fewer pixels per inch. So for an LS20 to scan at 73 pixels per inch, X averaging is automatically enabled.

2.2.5 Shifted Dynamic Range Mode

The Shifted Dynamic Range (SDR) mode is an option that changes the mapping of pixel values to optical density. Normally pixel values are equal to the optical density scaled by 1000. So an optical density of 1.24 is represented by a pixel value of 1240.

i.e. Pixel = OD * 1000

This allows a 12-bit pixel value to span the optical density range 0.0 to 4.0.

When the SDR mode is enabled the pixel mapping is offset by 1000 counts:

i.e. Pixel = (OD * 1000) - 1000

This allows a 12-bit pixel value to span the optical density range 1.0 to 5.0.

Currently, only the LS85-SDR model supports the SDR mode option.

2.2.6 User LUT

While scanning, the final step before putting a pixel into the DCB memory is to pass it through the User LUT. The User LUT is a 4096-entry, 16-bit lookup table supplied by the “user” or host application. By default, the User LUT is loaded with a “linear” table where the first entry is 0, the second entry is 1, and so on until the last entry is loaded with 4095. The User LUT can be used for anything. Normally it is used to invert the pixels to change the meaning of black and white. By default the digitizer produces pixel values that correspond to optical density. Larger values mean darker. For display on a CRT it is often desirable to “invert” the pixel values, making a zero pixel black. This can be done by loading an “inverted” user LUT.

The default LUT on the DACQ is a 1:1 table, i.e. LUT[0] = 0, LUT[1] = 1, ... LUT[4095] = 4095. The user has the ability to load his own LUT to cause varying effects on the image. For example, to invert the image the user would supply a LUT with "inverted" values, e.g. LUT[0] = 4095, LUT[1] = 4094, ... LUT[4095] = 0.

The user may scan with a "pixel depth" of 8 bits (SCAN_8BITS). If so, an 8-bit LUT should be downloaded to the DACQ. Each entry of the 8-bit LUT must be mapped into the LSB. A typical 8-bit LUT will be "step-like" in nature, e.g.

- LUT[0] .. LUT[15] = 0,
- LUT[16] .. LUT[31] = 1,
- ...
- LUT[4064] .. LUT[4079] = 254,
- LUT[4080] .. LUT[4095] = 255

NOTE: The previous 8-bit LUT "wastes" some of the available bits by mapping impossible Optical Densities into the 8 bits. A better method is to spread the pixel optical density across the 0 to 255 pixel values.

```
//  Get the Maximum Optical Density (maxOD) of the current digitizer

SHORT maxOpticalDensity;    // scaled by 1000, so OD=1.5 ==> 1500
DrvrInfoStruct DInfo;

    DInfo.infoType = DRVR_INFO_SYS_DATA;    // type of info to get
    DInfo.retBufferSize = sizeof(ScannerSystemDataStruct);
    if ((status = LS_GetDrvInfo(&DInfo)) == SUCCESS)
    { // ok, no error
        maxOpticalDensity = DInfo.infoBuffer.SystemInfo.Model.MaxOD;
    }

//  Create linear 8-bit User LUT mapping from 0 to maxOD into the
//  full 8 bits.  i.e. pixel OD=0 ==> 0, pixel OD= maxOD ==> 255
SHORT ix;
SHORT endOfVariablePartOfLUT;
SHORT LUT[ 4096];
SHORT capValue;
float stepValue;

    endOfVariablePartOfLUT = maxOpticalDensity;
    stepValue = 256.0 / (double) maxOpticalDensity;
    capValue = maxOpticalDensity / 16;  // convert 12-bit to 8-bit
    for (ix=0; ix <= endOfVariablePartOfLUT; ix++)
    { // create linear ramp from 0 to maxOD
        LUT[ix] = (SHORT) (stepValue * (float)ix);
    }
    for (ix=endOfVariablePartOfLUT; ix < 4096; ix++)
    { // fill in from maxOD to end of LUT
        LUT[ix] = capValue;
    }

//  now use LS_LoadLut() to load the "LUT" array into the digitizer
    status = LS_LoadLut(UNIVERSAL_USER_LUT_NUMBER, LUT);
```

2.2.7 6-Sheet Film Feeder

One of the film feeder options for the LSDT family of digitizers is an automatic 6-sheet feeder. The film feeder can load up to six sheets of film. The film feeder attaches to the front of the digitizer and provides six slots to hold film. The slots are numbered 1 through 6, counting from the back of the digitizer toward the front. Film is loaded in the order of the slot numbers. Slot 1 is special. Film placed in slot 1 will always be loaded into the digitizer. Slot 1 is the equivalent of the manual feed slot on printers.

The film feeder does NOT automatically load the film. The host application program is required to command the film feeder for each sheet of film. The LS_FeedFilm function must be called to activate the film feeder. When the LS_FeedFilm function is called, the next film slot is "opened". If the slot was empty, no film will be loaded.

The film feeder is reset manually by pressing a lever. The film feeder may be reset at any time. The host application can never tell the current position of the film feeder, or whether there is any film in the film feeder.

2.2.8 Bulk Autoloader Film Feeder

The other film feeder option for the LSDT family of digitizers is a bulk autoloader. This film feeder can load up to 100 sheets of film (fewer if they have barcodes attached). The film feeder attaches to the front of the digitizer and provides a single hopper in which to put film.

The film feeder does NOT automatically load the film. The host application program is required to command the film feeder for each sheet of film. The LS_FeedFilm function must be called to activate the film feeder. When the LS_FeedFilm function is called, the next film is picked up from the hopper and fed into the digitizer. If the hopper is empty, no film will be loaded. Unlike the 6-sheet feeder, the bulk autoloader requires that LS_FeedFilm be called for the first film as well as all the rest.

2.2.9 Barcode Reader

A barcode reader option is available on the LSDT family of digitizers. If present, the barcode reader is mounted to read barcodes as the film exits at the completion of a scan. Barcode data is valid from the time the barcode is read until the next scan is started (LS_StartFilmScan), however the barcode data can NOT be read until the digitizer has stopped scanning. The film eject does not have to complete first. A return value of HARDWARE_BUSY is returned if the digitizer is busy.

Example of reading the barcode data **after** all scan data has been read without error:

```
SHORT status;
CHAR barbuf[];
SHORT barLength;

barLength = sizeof(barbuf); // set length of buffer

//  loop until digitizer is NOT busy

while((status = LS_ReadBarCode(&barbuf, &barLng)) == HARDWARE_BUSY)
{
    barLength = sizeof(barbuf);     // reset length of buffer
    //  Note:   Every time LS_ReadBarCode is called barLength is changed
                kill some time, wait, do other things for ~1/4 second ...
}

if (status == 1)            // have VALID barcode data
{
    // barbuf[] has a 0 terminated string
    // barLng = strlen(barbuf)
}
else
if (status == 0)    // no barcode data was read
{
    //  so, nothing to do here
}
else
{
    //  "status" is an error code, use LS_ErrorString() to display
}
```

2.2.10 Film Present Sensor

A hardware film present sensor option is available on the LSDT family of digitizers. The film present sensor detects when film is loaded into the digitizer and is ready to be scanned. The sensor is independent of the film feeder option. The sensor does NOT detect film IN the film feeder, ONLY film ready to scan. When used with the film feeder,

- First, the sensor is tested to see if an actual sheet of film was loaded.

- If not, then command the film feeder to load the next film and then test the sensor again to see if an actual sheet of film was loaded.

One possible use of the film present sensor is for the application program to poll the sensor while waiting for film. When film is detected, the application can automatically start the scanning process. Call LS_isFilmPresent to test the state of the sensor.

2.3 Digitizer Capabilities

Model	Type	Spot Size (μ)	Max Density (OD)	Averaging Modes Supported	Minimum Film Width (inch)	Maximum Film Width (inch)
LS20 (low res)	CCD film	350	3.2	XY	7	14
LS20 (hi res)	CCD film	175	3.2	XY	7	14
LS50	Laser film	210	3.6	None	8	14
LS75 (older)	Laser film	100	3.6	X, XY	8	14
LS75 (newer)	Laser film	100	3.6	X, XY	8	14
LS85	Laser film	50	4.1	X, XY	8	10
LS85-LF	Laser film	100	4.1	X, XY	11	14
LS85-SDR	Laser film	50	5.1	X, XY	8	10
ACR-2000	Laser CR	87	3.2	X, XY	6	14

Model	Type	Minimum Pixels Per Inch	Maximum Pixels Per Inch	Minimum Pixels per Line	Maximum Pixels per Line	Scan Speed (bytes / sec)	Data Rate (bytes / sec)
LS20 (low res)	CCD film	73	146	256	1024	40	331,128
LS20 (hi res)	CCD film	146	146	256	2048	81	331,128
LS50	Laser film	36	128	256	1140	115	235,520
LS75 (older)	Laser film	55	256	512	2048	115	471,040
LS75 (newer)	Laser film	55	300	512	4096	115	942,080
LS85	Laser film	36	730	256	5120	75	768,000
LS85-LF	Laser film	36	400	256	4096	75	614,400
LS85-SDR	Laser film	36	730	256	5120	75	768,000
ACR-2000	Laser CR	73	585	512	5120	50	409,600

3. SOFTWARE FUNCTIONS

This section organizes the API functions by type or usage. These categories are only general descriptions. Refer to the function descriptions for the details of their use.

3.1 General Setup Functions

Setup functions are typically used before scanning is started.

LS_isFilmPresent	Returns the status of the film present sensor.
LS_MapImageWindow	Maps the DCB memory window into program address space.
LS_OpenDevice	Creates a handle to the LSDT device driver.
LS_ResetHardware	Resets the digitizer hardware and reinitializes the driver.
LS_ResetScan	Reinitializes the driver.
LS_SetDevNum	Selects one of multiple installed digitizers.

3.2 Special Operation Functions

Some of these functions change values where the defaults are normally used.

LS_GetApiInfo	Returns detailed version information of the library (LSDTAPI.DLL).
LS_GetDrvInfo	Returns driver information, superset of LS_GetStatus.
LS_GetVersion	Returns the version number of the library (LSDTAPI.DLL).
LS_LoadLut	Loads the specified hardware LUT.
LS_ReadLut	Reads the specified hardware LUT.
LS_SetNextScanParams	Sets special parameters for the next film scan.
LS_StopScanEject	Stops the current scan and ejects the film.

3.3 Scanning Functions

These routines are the main functions used to specify and obtain image data.

LS_FeedFilm	Tries to load the next film from film feeder hardware.
LS_MapImageData	Changes the offset of the image data mapped to the DCB window.
LS_ReadImageToPointer	Reads image data into application's memory.
LS_SetAveragingMode	Sets the averaging mode parameter for the next film scan.
LS_SetPMTvoltage	Sets the PMT bias voltage (changes the gain) for the next film scan.
LS_SetSDRMode	Sets the SDR mode for the next film scan.
LS_SetTrim	Sets how much should be trimmed (or added) to the film's width.
LS_StartFilmScan	Starts a film scan operation.

3.4 Post-Scan Functions

These functions are used after the scan operation completes.

LS_CloseDevice	Closes the handle to the LSDT device driver opened by LS_OpenDevice.
LS_GetDataCount	Returns the number of bytes of image data scanned into DCB memory.
LS_GetPixelsPerLine	Returns the pixels per line setting of the digitizer hardware.
LS_GetStatus	Returns general status information, subset of LS_GetDrvInfo.
LS_isBarCodeValid	Returns TRUE if barcode data is currently valid.
LS_ReadBarCode	Returns the data read by the optional barcode reader.
LS_UnMapImageWindow	Removes the DCB memory window from program address space.

3.5 Miscellaneous Functions

These functions are only used by some applications or while processing errors.

LS_DriverStateString	Returns ASCII string explanation for supplied driver state.
LS_ErrorString	Returns ASCII string explanation for supplied error code.
LS_GetLastScsiError	Returns Windows SCSI error (when using a SCSI digitizer).
LS_ModelNameString	Returns ASCII string explanation for supplied model number.
LS_PixelDepthString	Returns ASCII string explanation for supplied pixel depth code.
LS_PixelFormatString	Returns ASCII string explanation for supplied pixel format code.
LS_ScanModeString	Returns ASCII string explanation for supplied scan mode.
LS_StatusOfLastScanString	Returns ASCII string explanation for supplied status of last scan.

3.6 Diagnostic Functions

These functions are almost **never** used by applications. Use of these functions without a detailed knowledge of the digitizer operation and hardware will more than likely produce strange results. Many of these functions produce results that are dependent on the exact digitizer model.

These functions are only included for completeness; they are NOT recommended.

LS_ChangeTimer	Sets specified timer with specified value.
LS_DiagnosticScan	Starts diagnostic scanning.
LS_FileCopy	Copy files between host PC and SCSI digitizer.
LS_GetAllTimers	Reads all hardware timer values.
LS_GetDriverHandle	Gets the LSDT device driver's handle.
LS_GetTimer	Reads the specified timer value.
LS_Motor	Runs (forward or backward) or stops the film transport motor.
LS_ReadDiagnostic	Diagnostic read from DCB image memory, including invalid data.
LS_RegisterCmd	Direct read or write to DCB hardware register.
LS_ResetDriverVariables	Resets driver variables to their detect cylinder width state.
LS_SetLaserMode	Sets Laser mode or LED illumination control.
LS_StartFilmScanNoCLUT	Special scan without using the hardware CLUT.
LS_WriteMemory	Diagnostic write to DCB image memory.

3.7 Compatibility Functions

These functions have all been superseded by simpler or newer functions. They still function as documented, but we recommend using only the functions listed above.

LS_GetImageWindow	Was used for DOS, not needed in Windows.
LS_LoadCalibrationTable	Use LS_LoadLut (UNIVERSAL_CAL_LUT_NUMBER, …).
LS_LoadCorrectionLookupTable	Use LS_LoadLut (UNIVERSAL_CORR_LUT_NUMBER, …).
LS_LoadLookUpTable	Use LS_LoadLut (UNIVERSAL_USER_LUT_NUMBER, …).
LS_ReadCalibrationTable	Use LS_ReadLut (UNIVERSAL_CAL_LUT_NUMBER, …).
LS_ReadCorrectionLookupTable	Use LS_ReadLut (UNIVERSAL_CORR_LUT_NUMBER, …).
LS_ReadImageIntoBuffer	Use LS_ReadImageToPointer.
LS_ReadImageIntoFile	Use LS_ReadImageToPointer, write into file yourself.
LS_ReadImageIntoFile_Fast	Use LS_ReadImageToPointer, write into file yourself.
LS_ReadLookUpTable	Use LS_ReadLut (UNIVERSAL_USER_LUT_NUMBER, …).
LS_StartMotor	Use LS_Motor (FORWARD_ or REVERSE_MOTER_DIRECTION).
LS_StopMotor	Use LS_Motor (STOP_MOTOR_DIRECTION).

4. SOFTWARE APPLICATIONS

4.1 Linking LSDT Software Functions

To use the LSDT software functions:

1) ***Include "LS_TYPES.H", "LS_DT.H", and "LS_API.H" in your application*** (as in the code examples below)**.**

2) Link with the "LSDTAPI.LIB" library import file. Before you execute your application, make sure the LSDTAPI.DLL is in your PATH. Also, ensure that the device driver has been loaded.

4.2 Sample Code

In the LSDT software distribution there is a Windows example program in the DEMO subdirectory. This program, LSEXP, is a simple scanning example built using the Microsoft Visual C++ 1.52c (for Win16) and Visual C++ 5.0 (Win32) compiler tools. The program allows the scanning parameters to be set in a setup window and a film to be scanned into a file, using the specified parameters. The LSEXP program is intended to compile for both Win16 and Win32 by using conditional code based on the "_WIN32" symbol. After digitizing with LSEXP, the resulting image file is in the Lumisys image format and can be read by various Lumisys tools (e.g. PL.EXE).

The main scanning loop is driven by the Windows Timer Message. In addition to scanning the program shows examples of how to use the film present detector, how to use the film feeder, how to detect the digitizer model, and how to load the user LUT.

4.2.1 Very Simple Example

The following example will scan a film and create an image file. However during the scan the Windows message queue is NOT serviced so NOTHING can run until the scan completes. This code is acceptable if run in a separate thread. All of the real work is done by the LS_ReadImageIntoFile function. The resulting image file is in the LUMISYS format defined by the LS_HDR.H include file.

```
#include "LS_TYPES.H"       // standard LSDT data types
#include "LS_DT.H"          // standard LSDT defines
#include "LS_API.H"         // LSDT functions

SHORT status;

status = LS_OpenDevice();
if (status != SUCCESS)
    // error - driver not loaded or device in use ??
    exit(1);

//  code to start a film scan:
status = LS_StartFilmScan(1024, VARIABLE_RESOLUTION, SCAN_12BITS, LSB_FIRST);

//  check status
if (status == SUCCESS)
    // scan started successfully, Now read the image into a file
    status = LS_ReadImageIntoFile("IMAGE1.IMG");
else
    // start of scan failed, tell user ...

LS_CloseDevice();           // close digitizer
```

4.2.2 Better Example

A second example using the more common method where the application processes the image data one buffer at a time. This example uses the DCB memory mapping functions. By memory mapping, the application gets access to the image data directly without causing the LSDT driver to copy the data from the DCB to an application buffer. The application could still do the data copy, or the application could write directly from the DCB memory to a file. Note the DCB memory should be considered read-only, since writing to the DCB memory will have undefined results. One of the reasons you might want to read directly from the DCB memory is speed. The DCB interface is only 8-bit, so it is kind of slow.

NOTE: When using the SCSI interface, the memory mapping mechanism is "faked". Since the DCB is inside the digitizer, the operating system has no direct access to the DCB image memory.

This example is similar to the LSEXP program provided in the software distribution.

```
#include "LS_TYPES.H"       // standard LSDT data types
#include "LS_DT.H"          // standard LSDT defines
#include "LS_API.H"         // LSDT functions

SHORT status;

status = LS_OpenDevice();
if (status != SUCCESS)
{
    // Error! -- driver not loaded or device in use – do something!
    exit(1);
}

// Reset the digitizer hardware and driver to force a known state
status = LS_ResetHardware();
if (status)
{
    // Error! -- do something!
    exit(2);
}

// Set scanning parameters such as averaging mode (LS_SetAveragingMode)
// Now is the time to load a special "User" LUT if desired
if (you want to)
    LS_LoadLookUpTable(UNIVERSAL_USER_LUT_NUMBER, pLUT);

// If the digitizer is equipped with a film feeder
if ("time to feed film, i.e. not the first scan of batch)
{
    status = LS_FeedFilm();
    // at present LS_FeedFilm never has a error if the previous calls worked

    if (status != SUCCESS)
    {
    }
}
// Map DCB memory into address space
CHAR FAR *lpImageBuffer;    // pointer to DCB memory mapped to address space

    If ( LS_MapImageWindow((VOID FAR **)(&lpImageBuffer)) != SUCCESS)
    {   // Error could not map DCB
        wsprintf(szBuffer, "Failed to Map DCB memory - SCAN ABORTED");
        MessageBox(hWnd, szBuffer, szAppName, MB_OK | MB_ICONEXCLAMATION);
        exit(3);
    }
```

```
// Now is the time to Open an output file if desired
// start a film scan operation
status = LS_StartFilmScan(1024, VARIABLE_RESOLUTION, SCAN_12BITS, LSB_FIRST);

// check status
if (status != SUCCESS)      // Error !
{
    // do something - e.g. display an error message
    exit(4);
}

// Now the film scan process has started
ReadStruct *lpReadLine;     // pointer for reading image data struct
ReadStruct aStruct;
lpReadLine = &aStruct;
lpReadLine->imageOffset = 0;
lpReadLine->readSize = 0x8000;  // 32 KB
int DoneScanning = FALSE;       // set flag

while(!DoneScanning)
{
    // map the image data
    status = LS_MapImageData(lpReadLine);

    if (status == SUCCESS ||
       (status == END_OF_IMAGE && lpReadLine->bytesRead > 0) )
    {
        // Then we successfully read some (if not all) data
        // do something with the image data, write to file, search, ...

        if (status == END_OF_IMAGE)
        {
            DoneScanning = TRUE;
            status = SUCCESS;
        }
    } // end successful read of data
    else if (status == END_OF_IMAGE)    // but we didn't read any data
    {
        DoneScanning = TRUE;
        status = SUCCESS;
    }
    else if (status == NOT_ENOUGH_DATA) // nothing to do right now
    {
        // this is a good place to do something about Windows messages
        // if not in a separate thread
        // or just kill some time ...
        continue;   // don't update imageOffset
    }
    else if (status == HARDWARE_BUSY)   // nothing to do right now
    {
        continue;   // don't update imageOffset
    }
    else                                // unsuccessful read!!!
    {
        DoneScanning = TRUE;
        // status was set above, let it pass on
    }

    // update the "imageOffset" to account for the data just read
    lpReadLine->imageOffset += (long)(lpReadLine->bytesRead);
```

```
}
// end of scanning process

// the LS_UnMapImageWindow is very important in Win16, less so in Win32
LS_UnMapImageWindow();          // unmap DCB memory from address space

LS_CloseDevice();               // close digitizer
exit(0);
```

4.2.3 Finding Pixels Per Line

In many cases it is desirable to know the number of pixels per line for the current scan, before the scan is complete. When scanning in fixed resolution mode, or variable resolution mode with unknown film widths, the actual pixels per line is not known until the digitizer has detected the film size. So to find the actual pixels per line after the scan has started, check the driver status until the driver has found the film size and then call the LS_GetPixelsPerLine function. See the following:

```
SHORT GetActivePPL (VOID);      // function to determine when PPL is valid
SHORT status, pixels;

// Start the scan
if ((status = LS_StartFilmScan(1024, VARIABLE_RESOLUTION, SCAN_12BITS,
    LSB_FIRST)) != SUCCESS) // error
{
    abort !
}

while((pixels = GetActivePPL()) == 0)
{
    // digitizer has NOT found the film size yet, so
    // kill some time OR go away and come back later
}

if (pixels < 0) // error - scan failed
{
    abort !
}

// have valid Pixels per line

// ---------------------------------------------------------------------
// GetActivePPL - Get the Pixels Per Line for the current scan
// ---------------------------------------------------------------------

SHORT GetActivePPL(VOID)
{
    LS_GetStatus(&driverStatus);
    if (driverStatus.driverState >= SCAN_PHASE_8)
    {
        return LS_GetPixelsPerLine();   // have PPL
    }
    else
        if (driverStatus.statusOfLastScan != SCAN_IN_PROGRESS)
            return (-1);                // error

    return (0);                         // don't have PPL yet

}   // end of GetActivePPL
```

5. Library Functions Reference

The following is a list of ALL functions in the library in alphabetical order.

5.1 LS_ChangeTimer

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Changes one of the six timers on the DACQ.

```
SHORT LS_ChangeTimer (SHORT timerID, SHORT value);
```

Input	Description	Values
timerID	Identifies one of six DACQ timers	***SCAN_MOTOR_CLOCK_TIMER*** ***FILM_MOTOR_CLOCK_TIMER*** ***PIXEL_CLOCK_TIMER*** ***DELAY_TO_FIRST_PIXEL_TIMER*** ***PIXELS_PER_LINE_TIMER*** ***LINES_PER_IMAGE_TIMER***
value	New timer value	non-negative integer

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
INVALID_PARAMETER	Invalid parameter
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.2 LS_CloseDevice

Closes the handle to the device driver that was opened by LS_OpenDevice. This is a ***Win32*** function; it has no effect in ***Win16*** systems but is included there for compatibility (will return SUCCESS).

```
SHORT LS_CloseDevice (VOID);
```

Return Values	Description
SUCCESS	Function successful
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.3 LS_DiagnosticScan

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Commands the driver to start scanning in diagnostic mode. Some scan modes are without film.

```
SHORT LS_DiagnosticScan (SHORT diagScanMode,
                         SHORT pixelDepth,
                         SHORT pixelFormat);
```

Input	Description	Values
diagScanMode	defines the type of diagnostic scan to perform	***SCAN_MODE_1*** ***SCAN_MODE_2*** ***SCAN_MODE_3*** ***SCAN_MODE_4*** ***SCAN_MODE_5*** ***SCAN_MODE_6*** ***SCAN_MODE_7*** ***DIAG_SCAN_AND_DISPLAY_CONTINUOUS***
pixelDepth	8 or 12 bits per pixel	***SCAN_8BITS*** ***SCAN_12BITS***
pixelFormat	MSB or LSB first	***MSB_FIRST*** ***LSB_FIRST*** (normal PC setting)

Return Values	Description
SUCCESS	Function successful
PIXELS_PER_LINE_OUT_OF_RANGE	Invalid parameter
ILLEGAL_SCAN_MODE	Invalid parameter
ILLEGAL_BITS_PER_PIXEL	Invalid parameter
ILLEGAL_PIXEL_FORMAT	Invalid parameter
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.4 LS_DriverStateString

Returns the driver state string associated with the input driverState code value. **Be sure to supply an 80-character output buffer to receive the maximum string, including terminating NULL character.**

```
VOID LS_DriverStateString (SHORT driverState, CHAR FAR *strBuf);
```

Input	Description	Values
driverState	driver state code	value returned by LS_GetStatus – ***see LS_DT.H "Driver State DEFINES"***

Output	Description
strBuf	pointer to ASCII string describing driver state is copied into “strBuf” buffer

5.5 LS_ErrorString

Returns the error string associated with the input errorCode status. A user of this function may want to print out an ASCII error message for an error status received from some LSDT function. **Be sure to supply an 80-character output buffer to receive the maximum string, including terminating NULL character.**

```
VOID LS_ErrorString (SHORT errorCode, CHAR FAR *strBuf);
```

Input	Description	Values
errorCode	error ID code	***see LS_DT.H "Error DEFINES" & "Errors"***

Output	Description
strBuf	pointer to ASCII string describing supplied error code is copied into “strBuf” buffer

5.6 LS_FeedFilm

Commands the film feeder hardware to load the next film. What this does depends upon what film loading equipment the digitizer has:

- 6-sheet film feeder hardware — If it is armed (cocked) and film is present in the next slot, then the film will be dropped into position to be processed by the next scan operation.
- Bulk film loader — If there is film in it, then the film will be picked up and dropped into position to be processed by the next scan operation.
- No loading equipment — No effect.

```
SHORT LS_FeedFilm (VOID);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.7 LS_FileCopy

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

For SCSI interfaces only, transfers a disk file between the host PC and the digitizer. For DCB interfaces, SUCCESS is always returned.

```
SHORT LS_FileCopy (USHORT nSrcDev,     PCHAR lpszSrcFile,
                   USHORT nDstDev,     PCHAR lpszDstFile);
```

Input	Description	Values
nSrcDev	Source device (host PC, SCSI digitizer, etc.)	***see LS_DT.H***
lpszSrcFile	Source file name	***Any valid DOS path/filename, e.g. CLT12345.DAT or C:\WINDOWS\SYSTEM\CLT12345.DAT***
nDstDev	Source device (host PC, SCSI digitizer, etc.)	***see LS_DT.H***
lpszdstFile	Destination file name	***Any valid DOS path/filename, e.g. CLT12345.DAT or C:\WINDOWS\SYSTEM\CLT12345.DAT***

Return Values	Description
SUCCESS	Function successful
anything else	Error during file transfer

5.8 LS_GetAllTimers

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Returns the six DACQ timer values in a structure.

```
SHORT LS_GetAllTimers (GetTimerStruct FAR *timers);
```

Output	Description
timers	The structure is filled with the current values of the hardware timers. see LS_DT.H

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.9 LS_GetApiInfo

Returns a structure containing information regarding the library (LSDTAPI.DLL).

```
VOID LS_GetApiInfo (VerInfoStruct FAR *ApiInfo);
```

Output	Description
ApiInfo	The structure is filled with version information. see LS_DT.H

5.10 LS_GetDataCount

Reads the Data Count register from the driver, which indicates the number of bytes of image data scanned into DCB memory. When the LSDT is scanning a film, the low order 8 bits of the returned value are always zero.

NOTE: If the buffer has wrapped, the byte count returned will be larger than the amount of memory on the DCB.

```
LONG LS_GetDataCount (VOID);
```

Return Values	Description
Positive number	Number of scanned image bytes

5.11 LS_GetDriverHandle

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

For DCB interfaces only, returns the current driver handle. For SCSI interfaces, NULL will always be returned. This is a ***Win32*** function; it is not available in ***Win16***.

```
HANDLE LS_GetDriverHandle (VOID);
```

Return Values	Description
NULL	The digitizer driver is not installed
all other values	Driver handle

5.12 LS_GetDrvInfo

Returns dynamic and static status information from the driver. This is an expanded version of LS_GetStatus. Only up to retBufferSize bytes of data will be returned.

```
SHORT LS_GetDrvInfo (DrvrInfoStruct FAR *drvrInfo)
```

Input	Description	Values
drvrInfo	Pointer to a DrvrInfoStruct	***- see LS_DT.H***
->infoType	What type of status to return	***DRVR_INFO_GET_STATUS*** ***DRVR_INFO_HW_SETTINGS*** ***DRVR_INFO_LAST_SCAN*** ***DRVR_INFO_GET_LAST_TIMERS*** ***DRVR_INFO_GET_SYS_DATA*** ***DRVR_INFO_PRIV_DATA*** ***DRVR_INFO_AGC_DATA*** ***DRVR_INFO_EDGE_SCAN_PARAMS***
->retBufferSize	How many bytes should be returned: DRVR_INFO_GET_STATUS DRVR_INFO_HW_SETTINGS DRVR_INFO_LAST_SCAN DRVR_INFO_GET_LAST_TIMERS DRVR_INFO_GET_SYS_DATA DRVR_INFO_PRIV_DATA DRVR_INFO_AGC_DATA DRVR_INFO_ EDGE_SCAN_PARAMS	***sizeof(InfoStatusStruct)*** ***sizeof(HWSettingsStruct)*** ***sizeof(LastScanStruct)*** ***sizeof(GetTimerStruct)*** ***sizeof(ScannerSystemDataStruct)*** ***sizeof(ModelPrivateDataStruct)*** ***sizeof(DrvrInfoAGCStruct)*** ***sizeof(EdgeScanParamsStruct)***

Output	Description
drvrInfo	Pointer to a DrvrInfoStruct
->bytesReturned	Actual number of bytes returned (should be equal to what was requested)
->infoBuffer	DrvrInfoStruct – a union of several status structures

Return Values	Description
SUCCESS	Function successfully returned status
INVALID_BUFFER_SIZE	Invalid Return Buffer Size

5.13 LS_GetDrvVersion

Returns the version ID of the currently loaded driver. A NULL terminated string is returned; allow 30 bytes.

```
SHORT LS_GetDrvVersion (CHAR FAR *DrvVerString);
```

Output	Values	Description
DrvVerString	ex: "3.52 Build 12"	Pointer to ASCII string – Major (3), Minor (5), Maintenance (2), Build Number (12)

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.14 LS_GetImageWindow

NOT RECOMMENDED – OBSOLETE

Returns the physical address of the DCB Image Memory. This has no practical use for a Windows program. Here for DOS port compatibility only.

```
SHORT LS_GetImageWindow (VOID FAR * FAR *mapAddr)
```

Output	Description
mapAddr	Will contain the address of DCB image memory.

Return Values	Description
SUCCESS	Function successfully returned DCB Image Memory address

5.15 LS_GetLastSCSIError

For SCSI interfaces only, returns the error code associated with the last SCSI operation. For DCB interfaces, SUCCESS will always be returned.

```
DWORD LS_GetLastScsiError (VOID);
```

Return Values	Description
error code	See LS_DT.H and Appendix B in this manual

5.16 LS_GetPixelsPerLine

Returns the number of pixels per line of the most recently scanned image. Note: If you call this too early (before the driver has found the film edges and started scanning image data), the value returned is simply what you specified in the call to LS_StartFilmScan. It may very well NOT be what you are going to get in subsequent data. Wait until some image data is ready before calling this function.

```
SHORT LS_GetPixelsPerLine (VOID);
```

Return Values	Description
positive number	Number of pixels per line or ***DEFAULT_PIXELS_PER_LINE*** (1024) if no image has been scanned
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.17 LS_GetStatus

Returns the current status of the LSDT from the device driver. The status will be returned in the structure pointed to by the function argument. This is exactly the same as LS_GetDrvInfo() where drvrInfo->infoType is set to DRVR_INFO_GET_STATUS

```
SHORT LS_GetStatus (DriverStatusStruct FAR *driverStatus);
```

Output	Description
driverStatus	See LS_DT.H – location to store returned status

Return Values	Description
SUCCESS	Function successfully returned status
INVALID_PARAMETER	Invalid parameter
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.18 LS_GetTimer

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Returns one of the six timers on the DACQ.

```
SHORT LS_GetTimer (SHORT timerID);
```

Input	Description	Values
timerID	Identifies one of six DACQ timers	***SCAN_MOTOR_CLOCK_TIMER***
		FILM_MOTOR_CLOCK_TIMER
		PIXEL_CLOCK_TIMER
		DELAY_TO_FIRST_PIXEL_TIMER
		PIXELS_PER_LINE_TIMER
		LINES_PER_IMAGE_TIMER

Return Values	Description
Positive integer	Timer value for specified timer
INVALID_PARAMETER	Invalid parameter
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.19 LS_GetVersion

Returns a structure indicating the version number of the library (LSDTAPI.DLL).

```
VOID LS_GetVersion (VersionStruct FAR *version);
```

Output	Description
version	pointer to major and minor version number structure

5.20 LS_isBarCodeValid

Returns the status of the barcode data. A return value of 1 indicates that barcode data has become available since the last start of scan (LS_StartFilmScan). A return value of 0 indicates no barcode data is available or some kind of error (such as driver not loaded). This function makes use of the LS_GetDrvInfo function.

NOTE: Barcode data is valid from the time the barcode is read until the next scan is started; however, the barcode data can NOT be read until the digitizer has finished scanning data. If it is not available, the application program should wait until the digitizer is no longer in the "BUSY" state (i.e. while it is ejecting the film) and then try again (the barcode may be on the trailing edge of the film and not actually sensed until the eject phase).

```
SHORT LS_isBarCodeValid (VOID);
```

Return Values	Description
1 (***True***)	A valid barcode has been read since the last start of scan
0 (***False***)	No barcode data (or error)

5.21 LS_isFilmPresent

Returns the status of the Film Present detector. A return value of 1 indicates the presence of film. A return value of 0 indicates no film, no film present detector, or some kind of error (such as driver not loaded).

```
SHORT LS_isFilmPresent (VOID);
```

Return Values	Description
1 (***True***)	Detector sees film
0 (***False***)	No film (or error)

5.22 LS_LoadCalibrationTable

NOT RECOMMENDED – SUPERSEDED by LS_LoadLut

```
SHORT LS_LoadCalibrationTable (SHORT FAR *calibrationTable);
```

Input	Description	Values
calibrationTable	Location of Cal LUT values to load into the DACQ Cal LUT	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.23 LS_LoadCorrectionLookUpTable

NOT RECOMMENDED – SUPERSEDED by LS_LoadLut

```
SHORT LS_LoadCorrectionLookUpTable (SHORT FAR *LUT);
```

Input	Description	Values
LUT	Location of CLUT values to load into the DACQ CLUT	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.24 LS_LoadLookUpTable

NOT RECOMMENDED – SUPERSEDED by LS_LoadLut

```
SHORT LS_LoadLookUpTable (SHORT FAR *LUT);
```

Input	Description	Values
LUT	Location of User LUT values to load into the DACQ ULUT	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.25 LS_LoadLut

Writes a user-supplied Lookup Table into the specified LUT on the DACQ. The caller needs to be sure to supply a large enough buffer. The data format of each entry is LSB first. This is a generalized version of older, specific LUT functions: LS_LoadCalibrationTable, LS_LoadCorrectionLookUpTable, and LS_LoadLookUpTable.

NOTE: This function can only be done when the digitizer is IDLE.

UNIVERSAL_CAL_LUT_NUMBER **NOT RECOMMENDED – DIAGNOSTIC USE ONLY**

LS_LoadLut loads a Calibration Lookup Table (Cal LUT) into the Data Acquisition Board (DACQ) on the LSDT. The number of entries in the table is equal to the maximum number of pixels per line, which for some digitizers may be as large as 5120 16-bit entries.

UNIVERSAL_CORR_LUT_NUMBER **NOT RECOMMENDED – DIAGNOSTIC USE ONLY**

LS_LoadLut loads a Correction Lookup Table (CLUT) into the Data Acquisition Board (DACQ) on the LSDT. The CLUT on the DACQ contains 4096 16-bit entries.

UNIVERSAL_USER_LUT_NUMBER

LS_LoadLut loads a User Lookup Table (ULUT) into the Data Acquisition Board (DACQ) on the LSDT. The User LUT on the DACQ contains 4096 16-bit entries, into which the DACQ indexes with the 12-bit (0..4095) optical density value for each pixel scanned.

```
SHORT LS_LoadLut (USHORT LutNumber, SHORT FAR *LUT);
```

Input	Description	Values
LutNumber	LUT ID number	***UNIVERSAL_CAL_LUT_NUMBER*** ***UNIVERSAL_CORR_LUT_NUMBER*** ***UNIVERSAL_USER_LUT_NUMBER***
LUT	Pointer to LUT values to store into DACQ	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.26 LS_MapImageData

Commands the driver to select the image memory page specified by the imageOffset given. It is up to the application to read the image data from memory.

All the image data is accessed through a 32KB memory mapped window in the Memory Address Space of the ISA BUS. By factory default, the DCB is set up so that this window is at 0xD0000. It may be changed during installation. Data is always mapped on a 32KB boundary. To access a 10-MB image contained on this board, the application can read the data directly from the board but only 32KB at a time. After each 32KB chunk of image is read, the driver must be called to map to the next chunk of data. In the example file, LSEXP, the use of these calls (LS_MapImageWindow and LS_MapImageData) is demonstrated. This usage is very similar to LS_ReadImageToPointer.

NOTE: When using the SCSI interface, the memory mapping mechanism is "faked". Since the DCB is inside the digitizer, the operating system has no direct access to the DCB image memory.

```
SHORT LS_MapImageData (ReadStruct FAR *mapImageParam);
```

Input	Description	Values
mapImageParam	Pointer to Read Struct	***- see LS_DT.H***
->imageOffset	Offset from beginning of image data to set image memory map page.	***- see LS_DT.H***
->readSize	Number of bytes application wishes to map. Should be 32,768 (one full memory page).	***- see LS_DT.H***
->buffer[1]	Unused for this function.	***- see LS_DT.H***

Output	Description
mapImageParam	Pointer to Read Struct
->bytesRead	Number of bytes that application can access. If less than one full memory page, then application should continue sending **LS_MapImageData** commands until either 1) Full memory page is available for access, 2) End Of Image status received, or 3) Error or Scan Aborted status received. Application can then directly access image memory data safely.

Return Values	Description
SUCCESS	Updated image data requested is available in Image Memory
HARDWARE_FAILURE	Digitizer hardware malfunction
NOT_ENOUGH_DATA	Number of bytes available is smaller than the number of bytes to read – the available bytes will be read and that number of bytes will be stored in ->bytesRead. More scan data is expected
END_OF_IMAGE	End of Image Reached; data is still available if ->bytesRead > 0
SCAN_ABORTED	Scan aborted
HARDWARE_BUSY	Not in right scan state for this command

5.27 LS_MapImageWindow

Maps the DCB memory window into the program address space. Returns pointer to DCB Memory window.

NOTE: Windows has a limited number of resources to map memory. Be sure to call LS_UnMapImageWindow when not scanning.

NOTE: When using the SCSI interface, the memory mapping mechanism is "faked". Since the DCB is inside the digitizer, the operating system has no direct access to the DCB image memory.

```
SHORT LS_MapImageWindow (VOID FAR * FAR *imageWindowPtr);
```

Input	Description	Values
imageWindowPtr	Pointer to the 32KB DCB Memory window	virtual memory address

Return Values	Description
SUCCESS	Function successful
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.28 LS_ModelNameString

Returns the model name string associated with the input DACBModel model value. For example, an input value of 75 might be returned as "LS-75". You can obtain the model ID from LS_GetStatus or LS_GetDrvInfo. **Be sure to supply an 80-character output buffer to receive the maximum string, including terminating NULL character.**

```
VOID LS_ModelNameString (SHORT DACBModel, CHAR FAR *strBuf);
```

Input	Description	Values
DACBModel	Model ID number	e.g. 20, 50, 75, 76, 85, 86, 135

Output	Values	Description
strBuf	NULL terminated character string	ASCII text description of digitizer model

```
DriverStruct    ds;
CHAR            szMessage[80]

if (LS_GetStatus(&ds) == SUCCESS)
{
    LS_ModelNameString(ds.DACQModelRev.DACBModel, szMessage);
    printf("Digitizer Model: %s\n", szMessage);
}
```

5.29 LS_Motor

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Starts the film transport motor in the specified direction at the last motor speed or stops the motor. Use this generalized function rather than LS_StartMotor() and LS_StopMotor()

```
SHORT LS_Motor (SHORT motorCmd);
```

Input	Description	Values
motorCmd	Motor command code	***FORWARD_MOTOR_DIRECTION*** ***REVERSE_MOTOR_DIRECTION*** ***STOP_MOTOR_DIRECTION***

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
INVALID_PARAMETER	Invalid parameter
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.30 LS_OpenDevice

Creates a handle to the LSDT driver. This **MUST** be the first call to the LSDT API functions. This is a ***Win32*** function; it has no effect in ***Win16*** systems but can be included for compatibility (it always returns SUCCESS).

```
SHORT LS_OpenDevice (VOID);
```

Return Values	Description
SUCCESS	Function successful
DRIVER_NOT_INSTALLED	The digitizer driver is not installed
DEVICE_ALREADY_OPEN	Digitizer in use

5.31 LS_PixelDepthString

Returns the pixel depth string associated with the input pixelDepth code value. You can obtain the current pixel depth from LS_GetStatus. **Be sure to supply an 80-character output buffer to receive the maximum string, including terminating NULL character.**

```
VOID LS_PixelDepthString (SHORT pixelDepth, CHAR FAR *strBuf);
```

Input	Description	Values
PixelDepth	8-bit or 16-bit pixel specifier code	***SCAN_8BITS or SCAN_12BITS*** ***the pixelDepth value returned by LS_GetStatus – see LS_DT.H "Pixel Depth DEFINES"***

Output	Description
StrBuf	ASCII string describing supplied pixel depth code copied into “strBuf” buffer

5.32 LS_PixelFormatString

Returns the pixel format string associated with the input pixelFormat code value. You can obtain the current pixel format from LS_GetStatus. **Be sure to supply an 80-character output buffer to receive the maximum string, including terminating NULL character.**

```
VOID LS_PixelFormatString (SHORT pixelFormat, CHAR FAR *strBuf);
```

Input	Description	Values
pixelFormat	Pixel format code	***MSB_FIRST or LSB_FIRST*** ***the pixelFormat value returned by LS_GetStatus – see LS_DT.H "Pixel Format DEFINES"***

Output	Description
strBuf	ASCII string describing supplied pixel format is copied into “strBuf” buffer

5.33 LS_ReadBarCode

Returns the current barcode data, and a valid / not valid status. This function is called with a pointer to a buffer to receive the barcode data and a pointer to a length value. The length value defines the length of the user buffer on input, and on output the actual length of the returned barcode data. The maximum size of the barcode data is “LS_BARCODE_MAX_SIZE”, not including the terminating NULL character (“0”) which is appended to the return data if the user buffer is large enough. If the buffer is not large enough the terminating NULL will overwrite barcode data (i.e. the returned data always ends with a NULL character.) The return status is “1” for valid data, “0” with a return length of zero if the data is not valid; all other values are LSDT error codes.

NOTE: Barcode data is valid from the time the barcode is read until the next scan is started (LS_StartFilmScan), however the barcode data can NOT be read until the digitizer has stopped scanning. HARDWARE_BUSY is returned if the digitizer is busy. The data can be read during the film eject phase.

```
SHORT LS_ReadBarCode (CHAR FAR *barString, SHORT FAR *barStringSize);
```

Input	Description	Values
BarStringSize	Address of size of barString	***sizeof(barString);***

Output	Description	Values
BarStringSize	Contains length of the barString not counting the final 0 terminator	***strlen(barString);***
BarString	Location to store barcode data string	***NULL terminated string***

Return Values	Description
0	Valid barcode data
1	Barcode data NOT valid
HARDWARE_BUSY	Digitizer is still scanning, try again later
other values	ERROR!

5.34 LS_ReadCalibrationTable

NOT RECOMMENDED – SUPERSEDED by LS_ReadLut

```
SHORT LS_ReadCalibrationTable (SHORT FAR *calibrationTable);
```

Output	Description	Values
calibrationTable	Location to store DACQ Calibration Table values	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.35 LS_ReadCorrectionLookUpTable

NOT RECOMMENDED – SUPERSEDED by LS_ReadLut

```
SHORT LS_ReadCorrectionLookUpTable (SHORT FAR *LUT);
```

Output	Description	Values
LUT	Location to store DACQ Correction LUT values	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.36 LS_ReadDiagnostic

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

This function is very similar to the LS_ReadImageToPointer function. The principle difference is that the LS_ReadDiagnostic function reads memory even if there is no image data present.

```
SHORT LS_ReadDiagnostic (ReadStructPtr FAR *diagReadParamPtr);
```

Output	Description
diagReadParamPtr	Read Ptr Structure (see LS_DT.H)

Return Values	Description
SUCCESS	Function successfully read image data
INVALID_PARAMETER	Invalid parameter
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
HARDWARE_BUSY	This is returned when the digitizer is in a transitory state between the start of a scan and the actual scan. Try again!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.37 LS_ReadImageIntoBuffer

NOT RECOMMENDED – OBSOLETE

Commands the driver to read (copy) image data from the DCB memory into buffer memory on the host. This function is called once a scan starts. Typically, this function is called in a loop with readSize = 32K until an ***END_OF_IMAGE*** status is received.

This function uses an obsolete input parameter structure that defines parameter values in the same memory block as the returned scan data. Instead, you should use LS_ReadImageToPointer function with its "ReadStructPtr" struct.

```
SHORT LS_ReadImageIntoBuffer (ReadStruct FAR *readParam);
```

Input	Description	Values
readParam	Read Structure (see LS_DT.H)	a byte count (readSize) of 32,768 is recommended

Return Values	Description
SUCCESS	Function successfully read image data
INVALID_PARAMETER	Invalid parameter
NOT_ENOUGH_DATA	Number of bytes available is smaller than the number of bytes to read – the available bytes will be read and that number of bytes will be stored in ->bytesRead. More scan data is expected
END_OF_IMAGE	Same as ***NOT_ENOUGH_DATA*** except the digitizer has finished scanning the image
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
HARDWARE_BUSY	This is returned when the digitizer is in a transitory state between the start of a scan and the actual scan. Try again!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed
DATA_LOST	Either the image is larger than DCB memory or a data overrun error has occurred

5.38 LS_ReadImageIntoFile

NOT RECOMMENDED – OBSOLETE

Causes the driver to read (copy) an entire image from the DCB memory into a file on the host. As the data becomes available from the DCB, it is written into a host file. This file is in Lumisys image format (see Appendix A).

NOTE: It will ignore the Windows message queue for however long it takes to write the file (which is why this is not recommended). It calls LS_ReadImageToPointer().

```
SHORT LS_ReadImageIntoFile (CHAR FAR *fileName );
```

Input	Description	Values
fileName	path \ filename	Any valid DOS path \ filename, e.g. XRAY1.IMG or C:\USERS\JACK\XRAY1.IMG

Return Values	Description
SUCCESS	Function successfully read image into file
FILE_OPEN_ERROR	Could not open file
FILE_WRITE_ERROR	Could not write any or all bytes to file
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
MEMORY_ALLOCATION_FAILURE	Could not allocate enough memory
DRIVER_NOT_INSTALLED	The digitizer driver is not installed
DATA_LOST	Either the image is larger than DCB memory or a data overrun error has occurred

5.39 LS_ReadImageIntoFile_Fast

NOT RECOMMENDED – OBSOLETE

This function is the same as the LS_ReadImageIntoFile function, except implemented using the DCB mapping functions of the driver to read the incoming image directly from DCB memory. In the Win32 version of the library, this function is exactly the same as LS_ReadImageIntoFile().

```
SHORT LS_ReadImageIntoFile_Fast (CHAR FAR *fileName );
```

Input	Description	Values
FileName	path \ filename	***Any valid DOS path/filename, e.g. XRAY1.IMG or C:\USERS\JACK\XRAY1.IMG***

Return Values	Description
SUCCESS	Function successfully read image into file
FILE_OPEN_ERROR	Could not open file
FILE_WRITE_ERROR	Could not write any or all bytes to file
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
MEMORY_ALLOCATION_FAILURE	Could not allocate enough memory
DRIVER_NOT_INSTALLED	The digitizer driver is not installed
DATA_LOST	Either the image is larger than DCB memory or a data overrun error has occurred

5.40 LS_ReadImageToPointer

Commands the driver to read (copy) image data from the DCB memory into buffer memory on the host. This function may be called when a scan is occurring. An application will typically call this function in a loop with ->readSize = 32K until an ***END_OF_IMAGE*** status is received. This function differs from LS_ReadImageIntoBuffer in that a field of the ReadStructPtr structure contains a FAR pointer specifying where to store the image data – as opposed to the buffer starting in the ReadStruct structure itself.

```
SHORT LS_ReadImageToPointer (ReadStructPtr FAR *readParam);
```

Input	Description	Values
readParam	Read Ptr Structure (see LS_DT.H)	a byte count (readSize) of 32,768 is recommended

Return Values	Description
SUCCESS	Function successfully read image data
INVALID_PARAMETER	Invalid parameter
NOT_ENOUGH_DATA	Number of bytes available is smaller than the number of bytes to read – the available bytes will be read and that number of bytes will be stored in ->bytesRead. More scan data is expected
END_OF_IMAGE	Same as ***NOT_ENOUGH_DATA*** except the digitizer has finished scanning the image
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
HARDWARE_BUSY	This is returned when the digitizer is in a transitory state between the start of a scan and the actual scan. Try again!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed
DATA_LOST	Either the image is larger than DCB memory or a data overrun error has occurred

5.41 LS_ReadLookUpTable

NOT RECOMMENDED – SUPERSEDED by LS_ReadLut

```
SHORT LS_ReadLookUpTable (SHORT FAR *LUT);
```

Output	Description	Values
LUT	Location to store DACQ ULUT values	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.42 LS_ReadLut

Reads the specified LUT from the DACQ into a user-supplied buffer. The caller needs to be sure to supply a large enough buffer (up to 10,240 bytes). The data format of each entry is LSB first. This is a generalized version of older, specific and now obsolete LUT functions: LS_ReadCalibrationTable, LS_ReadCorrectionLookUpTable, and LS_ReadLookUpTable.

NOTE: This function can only be done when the digitizer is IDLE.

UNIVERSAL_CAL_LUT_NUMBER

LS_ReadLut reads the Calibration Lookup Table (Cal LUT) currently stored in the DACQ. Each entry in the table is an unsigned 16-bit integer. The number of entries in the table is equal to the number of pixels per line. The caller needs to be sure to supply a large enough buffer, 2 bytes for each pixel for the digitizer's maximum line size.

UNIVERSAL_CORR_LUT_NUMBER

LS_ReadLut reads the 4096 2-byte entries from the Correction Lookup Table (CLUT) currently stored in the DACQ.

UNIVERSAL_USER_LUT_NUMBER

LS_ReadLut reads the 4096 2-byte entries from the user Lookup Table (ULUT) currently stored in the DACQ. The User LUT on the DACQ contains 4096 16-bit entries, into which the DACQ indexes with the 12-bit (0..4095) optical density value for each pixel scanned.

```
SHORT LS_ReadLut (USHORT LutNumber, SHORT FAR *LUT);
```

Input	Description	Values
LutNumber	LUT ID number	***UNIVERSAL_CAL_LUT_NUMBER***
		UNIVERSAL_CORR_LUT_NUMBER
		UNIVERSAL_USER_LUT_NUMBER

Output	Description	Values
LUT	Location to store DACQ LUT values	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.43 LS_RegisterCmd

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Reads or writes directly to the digitizer hardware registers.

```
SHORT LS_RegisterCmd (RegDataCmdStruct FAR *RegDataCmd);
```

Input / Output	Description
RegDataCmd	See LS_DT.H

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.44 LS_ResetDriverVariables

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Resets the digitizer driver scanning parameters to their initial settings used to detect cylinder width. This function is typically only used in diagnostics.

```
SHORT LS_ResetDriverVariables (VOID);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.45 LS_ResetHardware

Resets the LSDT DACQ hardware and reinitializes the LSDT digitizer driver. This function is a superset of the LS_ResetScan function. The digitizer LUTs are reset to their initial state at the time the driver was loaded in addition to the LS_ResetScan operations (see LS_ResetScan).

```
SHORT LS_ResetHardware (VOID);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.46 LS_ResetScan

Reinitializes the LSDT digitizer driver. It does NOT reset the DACQ hardware (the DACQ retains its current Lookup and Calibration Tables). Stops the current scan if one is in progress. The LS_ResetScan functionality is automatically performed at the end of all normal scans, when a start scanning (LS_StartFilmScan) fails due to bad parameters, and when the edges of the film can not be found during a scan operation.

NOTE: The film is not ejected.

```
SHORT LS_ResetScan (VOID);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.47 LS_ScanModeString

Returns the scan mode string associated with the input scanMode code value. You can obtain the scan mode from LS_GetStatus. **Be sure to supply an 80-character output buffer to receive the maximum string, including terminating NULL character.**

```
VOID LS_ScanModeString (SHORT scanMode, CHAR FAR *strBuf);
```

Input	Description	Values
scanMode	Scan mode value code	***scanMode value returned by LS_GetStatus – see LS_DT.H "Scan Mode DEFINES"***

Output	Description
strBuf	Pointer to ASCII string describing supplied scan mode – "Variable resolution" or " Fixed resolution"

5.48 LS_SetAveragingMode

Directs the digitizer driver to use the specified pixel averaging mode (see section 2.2.4) during the next scan only, regular or diagnostic.

```
SHORT LS_SetAveragingMode (SHORT averagingMode);
```

Input	Description	Values
averagingMode	Specifies pixel averaging mode	***NO_AVERAGING***
		X_ONLY_AVERAGING
		X_AND_Y_AVERAGING

Return Values	Description
SUCCESS	Function successful
ILLEGAL_AVG_MODE	Illegal averaging mode

5.49 LS_SetDevNum

Allows you to select a particular DCB if you have more than one installed. Only useful for Win16.

```
SHORT LS_SetDevNum (USHORT targetDevNum);
```

Input	Description	Values
targetDevNum	Which board	1, 2, 3, …

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.50 LS_SetLaserMode

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Directs the DACQ to set the scanning laser (or CCD illumination sub-system) to the specified mode.

```
SHORT LS_SetLaserMode (SHORT laserMode);
```

Input	Description	Values
laserMode	Specifies laser mode	***LASER_OFF***
		LASER_IDLE
		LASER_ON

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.51 LS_SetNextScanParams

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Sets special parameters for the next film scan or Mode 5 Diagnostic Scan. All values are reset to their default values after the next scan completes. This function ONLY affects the next scan.

```
SHORT LS_SetNextScanParams (SetNextScanFilmStruct FAR *NextScanParams);
```

Input	Description	Values
NextScanParams	Pointer to a Parameter Block	
->ParamType	Parameter type	***NEXT_SCAN_TYPE_PMT_VOLT*** ***NEXT_SCAN_TYPE_AUTOZERO*** ***NEXT_SCAN_TYPE_ENDSCAN*** ***NEXT_SCAN_TYPE_LASER*** ***NEXT_SCAN_TYPE_SDR*** ***NEXT_SCAN_TYPE_NOEDGE*** ***NEXT_SCAN_TYPE_AVG_MODE***
->ParamValue	Value to set	

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.52 LS_SetPMTvoltage

Allows you to adjust the PMT high-voltage bias. This function ONLY affects the next scan. CR digitizers only.

```
SHORT LS_SetPMTvoltage (UCHAR PMTadjust);
```

Input	Description	Values
PMTadjust	How much to adjust PMT voltage	***all values from 0 to 255, inclusive:*** ***0*** ***max Neg. voltage (less gain)*** ***128*** ***0V (nominal or default gain)*** ***255*** ***max Pos. voltage (more gain)***

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.53 LS_SetSDRMode

Enables the Shifted Dynamic Range (SDR) mode for the next scan. After the next scan, the mode is automatically disabled. This is only used with an LS85 with the SDR option.

```
SHORT LS_SetSDRMode (int SdrMode);
```

Input	Description	Values
SdrMode	Turn SDR mode ON or OFF	***SDR_MODE_ENABLE*** ***SDR_MODE_DISABLE***

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.54 LS_SetTrim

Adds or subtracts pixels from where the film edges are detected. Its primary purpose is to account for the rounded corners on scanned media. You can also trim a bit extra to insure that no "air" is included on the image edges. This function ONLY affects the next scan.

```
SHORT LS_SetTrim (SHORT leftTrim, SHORT rightTrim);
```

Input	Description
leftTrim	Number of mils (.001") to trim from (if positive) or add to (if negative) left edge of film
rightTrim	Number of mils (.001") to trim from (if positive) or add to (if negative) right edge of film

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.55 LS_StartFilmScan

Commands the driver to start scanning a film.

```
SHORT LS_StartFilmScan (SHORT pixelResolution, SHORT scanResMode,
                        SHORT pixelDepth,      SHORT pixelFormat);
```

Input	Description	Values
pixelResolution	Pixels per line (scanMode = VARIABLE) Pixels per inch (scanMode = FIXED)	***256 ... 5120 – depends on digitizer model*** ***36 ... 730 – depends on digitizer model***
scanResMode	FIXED or VARIABLE mode	***FIXED_RESOLUTION*** ***VARIABLE_RESOLUTION***
pixelDepth	8 or 12 bits per pixel	***SCAN_8BITS*** ***SCAN_12BITS***
pixelFormat	MSB or LSB first	***MSB_FIRST*** ***LSB_FIRST (normal PC setting)***

Return Values	Description
SUCCESS	Function successful
PIXELS_PER_LINE_OUT_OF_RANGE	Invalid parameter
ILLEGAL_SCAN_MODE	Invalid parameter
ILLEGAL_BITS_PER_PIXEL	Invalid parameter
ILLEGAL_PIXEL_FORMAT	Invalid parameter
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.56 LS_StartFilmScanNoCLUT

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Commands the driver to start scanning a film. This function differs from the LS_StartFilmScan function because the pixel data bypasses the "CLUT" hardware. The CLUT corrects the pixel data value to the true optical density. LS_StartFilmScanNoCLUT is only used by diagnostic programs.

```
SHORT LS_StartFilmScanNoCLUT (SHORT pixelResolution,    SHORT
scanResMode,
                              SHORT pixelDepth,         SHORT
pixelFormat);
```

Input	Description	Values
pixelResolution	Pixels per line (scanMode = VARIABLE) Pixels per inch (scanMode = FIXED)	***256 ... 5120 -- depends on digitizer model*** ***36 ... 730 -- depends on digitizer model***
scanResMode	FIXED or VARIABLE mode	***FIXED_RESOLUTION*** ***VARIABLE_RESOLUTION***
pixelDepth	8 or 12 bits per pixel	***SCAN_8BITS*** ***SCAN_12BITS***
pixelFormat	MSB or LSB first	***MSB_FIRST*** ***LSB_FIRST (normal PC setting)***

Return Values	Description
SUCCESS	Function successful
PIXELS_PER_LINE_OUT_OF_RANGE	Invalid parameter
ILLEGAL_SCAN_MODE	Invalid parameter
ILLEGAL_BITS_PER_PIXEL	Invalid parameter
ILLEGAL_PIXEL_FORMAT	Invalid parameter
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.57 LS_StartMotor

NOT RECOMMENDED – DIAGNOSTIC USE ONLY – SUPERSEDED by LS_Motor

Starts the film transport motor in the specified direction at the last specified motor speed.

```
SHORT LS_StartMotor (SHORT direction);
```

Input	Description	Values
Direction	Direction to start motor	***FORWARD_MOTOR_DIRECTION*** ***REVERSE_MOTOR_DIRECTION***

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
INVALID_PARAMETER	Invalid parameter
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.58 LS_StatusOfLastScanString

Returns the status of last scan string associated with the input statusOfLastScan code value. You can obtain the status of last scan from LS_GetStatus. **Be sure to supply an 80-character output buffer to receive the maximum string, including terminating NULL character.**

```
VOID LS_StatusOfLastScanString (SHORT statusOfLastScan, CHAR FAR *strBuf);
```

Input	Description	Values
statusOfLastScan	Status code value	***statusOfLastScan value returned by LS_GetStatus function – see LS_DT.H "Status of Last Scan DEFINES"***

Output	Description
strBuf	Pointer to ASCII string describing supplied status code is copied into “strBuf” buffer

5.59 LS_StopMotor

NOT RECOMMENDED – DIAGNOSTIC USE ONLY – SUPERSEDED by LS_Motor

Stops the film transport motor in the LSDT.

```
SHORT LS_StopMotor (VOID);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.60 LS_StopScanEject

Stops the current film scan and ejects the film.

```
SHORT LS_StopScanEject (VOID);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.61 LS_UnMapImageWindow

Removes the DCB Memory window from the user's address space. See LS_MapImageWindow.

NOTE: Windows has a limited number of resources to map memory. Be sure to call LS_UnMapImageWindow when not scanning.

NOTE: When using the SCSI interface, the memory mapping mechanism is "faked". Since the DCB is inside the digitizer, the operating system has no direct access to the DCB image memory.

```
SHORT LS_UnMapImageWindow (VOID);
```

Return Values	Description
SUCCESS	Function successfully freed address space
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.62 LS_WriteMemory

NOT RECOMMENDED

Writes data into the DCB image memory. This function is used to perform memory tests. This function is ONLY available in the Win32 version of the library.

```
SHORT LS_WriteMemory (WriteParamStruct *writeImageParamPtr);
```

Input	Description
writeImageParamPtr	see LS_DT.H

Return Values	Description
SUCCESS	Function successful

Appendix A: LUMISYS IMAGE HEADER FORMAT

Byte Offset	Type	Description	Nr Bytes	Max Bytes	Example	Notes
0 – 803	N/A	Internal Use	804			* DO NOT USE! *
804 – 805	Integer	**Number of Images**	2		1	
806 – 807	Integer	**Pixels per Line**	2		2048	
808 – 809	Integer	**Lines per Image**	2		2486	
810 – 811	Integer	**Bits per Pixel**	2		12	
812 – 813	Integer	**Window**	2		4095	0 or 65535 imply not used
814 – 815	Integer	**Level**	2		2048	0 or 65535 imply not used
816 – 830	String	**Filename**	15	14	"LSDT.IMG"	
831 – 844	String	**Date**	14	13	"08/12/99"	
845 – 848	Long	**Time**	4		875029872	DOS Time Format (Seconds since 01/01/1970)
849 – 878	String	**Image Source**	30	29	"img_device"	Where image came from. Could be person, company, etc.
879 – 958	String	**Comment**	80	79	"Filtered & enhanced"	
959 – 989	String	**System Description**	31	30	"LS75"	
990 – 999	String	Header ID	10	N/A	"Lumisys"	Header format authentication (Exactly as shown!)
1000 – 1007	String	Version ID	8	N/A	"Hdr_Ver"	Version authentication (Exactly as shown!)
1008 – 1009	Integer	Version Number	2		4	Signifies version 4
1010 – 1011	Integer	**Byte Order**	2		0	0 = LSB, 1 = MSB, 2 = ?SB
1012 – 2047	N/A	Internal Use	1036			* DO NOT USE! *

These values are accessible in the include files LS_HDR.H and LS_HDROB.H. The latter is more object oriented and can be used with LS_HDROB.C. Support for images is supplied in LS_IMG.H and LS_IMG.C.

All header data is stored LSB first; the "Byte Order" field only pertains to the image data which follows.

"Max Bytes" means how many bytes you can put into this string field, not counting the terminating NULL character. Be sure to terminate all strings with a 0.

In general, for pre-existing headers, you should only change the bold-faced items above ("Number of Images" to "System Description").

If "Hdr_Ver" is not found at 1000, then the Version Number at 1008 is undefined.

If creating a header from scratch, the Header ID, Version ID and Version Number are required, in addition to reasonable values of all the bold-faced items.

Version 3 Added Version ID and Version Number
Version 4 Added Byte Order

Appendix B: FUNCTION ERROR CODES

This section contains a complete list of possible decimal error codes returned by LSDT functions. The corresponding #defines are in LS_DT.H. If the meaning below is blank, then that error code is not used.

Code	*Meaning*	*Code*	*Meaning*	*Code*	*Meaning*
0	No error – successful	26	Illegal averaging mode	51	Map Window error
1	Illegal parameter buffer size	27	Illegal Bytes To Write	52	Window Not Mapped
2	Illegal driver function code	28	Data Lost (a)	53	Unmap Window error
3	Pixels per line out of range	29	SDR shift error	54	DLL Driver Rev Mismatch
4	Illegal scan mode	30	Multiple File Matches	55	Unknown Debug Subcode
5	Illegal bits per pixel	31	Bad File Pathname	56	Window Already Mapped
6	Illegal pixel format	32	Open File error	57	Illegal Image Offset
7	Illegal bytes to read	33	Read File error	58	SCSI General error
8	Hardware busy	34	Close File error	59	SCSI Device Not Present
9	Scan Timeout error	35	Illegal Parameter	60	SCSI LSDT Scanner Not Found
10		36		61	SCSI Inquiry Response Format Invalid
11	Not enough data	37	Cylinder Edge Detect Threshold Error	62	SCSI Sense error
12	End of image	38		63	SCSI Scanner Not Open
13		39	Film Edge Detect Bad Offset	64	Function Not Supported
14	Calibration table out of range	40	Film Edge Detect No Film	65	File Open error
15		41		66	File Write error
16		42	Film Edge Detect Bad Limits	67	File Close error
17		43	Special Functions Disabled (b)	68	File Read error
18		44	Film Edge Detect Bad Edges (Edge detect failed)	69	File Size error
19				70	File Disk Space error
20		45	Illegal Pixels Per Inch (c)	71	File Not Specified
21	Hardware failure	46	Fixed Resolution error (d)	72	End Of File
22	Scan aborted error	47	Unknown error	73	Not Enough Disk Space
23	CLUT file error	48	Memory Allocation error	74	Invalid Filename
24	LUT file size error	49	Device Already Open	75	No Lumisys Image Header Found
25		50	Driver Not Installed		

(a) Either the image is larger than the available memory on the DCB or in the “wrapping” mode the application has not taken the data fast enough to keep up with the incoming scan data. Pedal faster.

(b) Some functions can only be called in debug mode.

(c) For example, an LS-20 allows only 73 or 146 DPI

(d) You specified a value that will chop off or add more than 1”.

37 SCSI ERROR CODES

This section contains a complete list of possible hexadecimal error codes returned by LS_GetLastScsiError().

NOTE: Most of these error codes will not be returned by LSDT digitizers.

SCSI Error Codes – Numerical Order			
0x0000	SCSI_GOOD	0x0001	FILEMARK_DETECTED
0x0002	END_OF_PARTITION_MEDIUM_DETECTED	0x0003	SETMARK_DETECTED
0x0004	BEGINNING_OF_PARTITION_MEDIUM_DETECTED	0x0005	END_OF_DATA_DETECTED
0x0006	I_O_PROCESS_TERMINATED	0x0011	AUDIO_PLAY_OPERATION_IN_PROGRESS
0x0012	AUDIO_PLAY_OPERATION_PAUSED	0x0013	AUDIO_PLAY_OPERATION_SUCCESSFULLY_COMPLETED
0x0014	AUDIO_PLAY_OPERATION_STOPPED_DUE_TO_ERROR	0x0015	NO_CURRENT_AUDIO_STATUS_TO_RETURN
0x0100	NO_INDEX_SECTOR_SIGNAL	0x0200	NO_SEEK_COMPLETE
0x0300	PERIPHERAL_DEVICE_WRITE_FAULT	0x0301	NO_WRITE_CURRENT
0x0302	EXCESSIVE_WRITE_ERRORS	0x0400	LOGICAL_UNIT_NOT_READY_CAUSE_NOT_REPORTABLE
0x0401	LOGICAL_UNIT_IS_IN_PROCESS_OF_BECOMING_READY	0x0402	LOGICAL_UNIT_NOT_READY_INITIALIZING_COMMAND_REQUIRED
0x0403	LOGICAL_UNIT_NOT_READY_MANUAL_INTERVENTION_REQUIRED	0x0404	LOGICAL_UNIT_NOT_READY_FORMAT_IN_PROGRESS
0x0500	LOGICAL_UNIT_DOES_NOT_RESPOND_TO_SELECTION	0x0600	REFERENCE_POSITION_FOUND
0x0700	MULTIPLE_PERIPHERAL_DEVICES_SELECTED	0x0800	LOGICAL_UNIT_COMMUNICATION_FAILURE
0x0801	LOGICAL_UNIT_COMMUNICATION_TIME_OUT	0x0802	LOGICAL_UNIT_COMMUNICATION_PARITY_ERROR
0x0900	TRACK_FOLLOWING_ERROR	0x0901	TRACKING_SERVO_FAILURE
0x0902	FOCUS_SERVO_FAILURE	0x0903	SPINDLE_SERVO_FAILURE
0x0A00	ERROR_LOG_OVERFLOW	0x0C00	WRITE_ERROR
0x0C01	WRITE_ERROR_RECOVERED_WITH_AUTO_REALLOCATION	0x0C02	WRITE_ERROR_AUTO_REALLOCATION_FAILED
0x1000	ID_CRC_OR_ECC_ERROR	0x1100	UNRECOVERED_READ_ERROR
0x1101	READ_RETRIES_EXHAUSTED	0x1102	ERROR_TOO_LONG_TO_CORRECT
0x1103	MULTIPLE_READ_ERRORS	0x1104	UNRECOVERED_READ_ERROR_AUTO_REALLOCATE_FAILED
0x1105	L_EC_UNCORRECTABLE_ERROR	0x1106	CIRC_UNRECOVERED_ERROR
0x1107	DATA_RESYNCHRONIZATION_ERROR	0x1108	INCOMPLETE_BLOCK_READ
0x1109	NO_GAP_FOUND	0x110A	MISCORRECTED_ERROR
0x110B	UNRECOVERED_READ_ERROR_RECOMMEND_REASSIGNMENT	0x110C	UNRECOVERED_READ_ERROR_RECOMMEND_REWRITE_THE_DATA
0x1200	ADDRESS_MARK_NOT_FOUND_FOR_ID_FIELD	0x1300	ADDRESS_MARK_NOT_FOUND_FOR_DATA_FIELD
0x1400	RECORDED_ENTITY_NOT_FOUND	0x1401	RECORD_NOT_FOUND
0x1402	FILEMARK_OR_SETMARK_NOT_FOUND	0x1403	END_OF_DATA_NOT_FOUND
0x1404	BLOCK_SEQUENCE_ERROR	0x1500	RANDOM_POSITIONING_ERROR
0x1501	MECHANICAL_POSITIONING_ERROR	0x1502	POSITIONING_ERROR_DETECTED_BY_READ_OF_MEDIUM
0x1600	DATA_SYNCHRONIZATION_MARK_ERROR	0x1700	RECOVERED_DATA_WITH_NO_ERROR_CORRECTION_APPLIED
0x1701	RECOVERED_DATA_WITH_RETRIES	0x1702	RECOVERED_DATA_WITH_POSITIVE_HEAD_OFFSET
0x1703	RECOVERED_DATA_WITH_NEGATIVE_HEAD_OFFSET	0x1704	RECOVERED_DATA_WITH_RETRIES_AND_OR_CIRC_APPLIED
0x1705	RECOVERED_DATA_USING_PREVIOUS_SECTOR_ID	0x1706	RECOVERED_DATA_WITHOUT_ECC_DATA_AUTO_REALLOCATED
0x1707	RECOVERED_DATA_WITHOUT_ECC_RECOMMEND_REASSIGNMENT	0x1708	RECOVERED_DATA_WITHOUT_ECC_RECOMMEND_REWRITE
0x1800	RECOVERED_DATA_WITH_ERROR_CORRECTION_APPLIED	0x1801	RECOVERED_DATA_WITH_ERROR_CORRECTION_AND_RETRIES_APPLIED
0x1802	RECOVERED_DATA_DATA_AUTO_REALLOCATED	0x1803	RECOVERED_DATA_WITH_CIRC
0x1804	RECOVERED_DATA_WITH_LEC	0x1805	RECOVERED_DATA_RECOMMEND_REASSIGNMENT
0x1806	RECOVERED_DATA_RECOMMEND_REWRITE	0x1900	DEFECT_LIST_ERROR
0x1901	DEFECT_LIST_NOT_AVAILABLE	0x1902	DEFECT_LIST_ERROR_IN_PRIMARY_LIST
0x1903	DEFECT_LIST_ERROR_IN_GROWN_LIST	0x1A00	PARAMETER_LIST_LENGTH_ERROR

SCSI Error Codes – Numerical Order

0x1B00	SYNCHRONOUS_DATA_TRANSFER_ERROR	0x1C00	DEFECT_LIST_NOT_FOUND
0x1C01	PRIMARY_DEFECT_LIST_NOT_FOUND	0x1C02	GROWN_DEFECT_LIST_NOT_FOUND
0x1D00	MISCOMPARE_DURING_VERIFY_OPERATION	0x1E00	RECOVERED_ID_WITH_ECC
0x2000	INVALID_COMMAND_OPERATION_CODE	0x2100	LOGICAL_BLOCK_ADDRESS_OUT_OF_RANGE
0x2101	INVALID_ELEMENT_ADDRESS	0x2200	ILLEGAL_FUNCTION
0x2400	INVALID_FIELD_IN_CDB	0x2500	LOGICAL_UNIT_NOT_SUPPORTED
0x2600	INVALID_FIELD_IN_PARAMETER_LIST	0x2601	PARAMETER_NOT_SUPPORTED
0x2602	PARAMETER_VALUE_INVALID	0x2603	THRESHOLD_PARAMETERS_NOT_SUPPORTED
0x2700	WRITE_PROTECTED	0x2800	NOT_READY_TO_READY_TRANSITION
0x2801	IMPORT_OR_EXPORT_ELEMENT_ACCESSED	0x2900	POWER_ON_RESET_OR_BUS_DEVICE_RESET_OCCURRED
0x2A00	PARAMETERS_CHANGED	0x2A01	MODE_PARAMETERS_CHANGED
0x2A02	LOG_PARAMETERS_CHANGED	0x2B00	COPY_CANNOT_EXECUTE_SINCE_HOST_CANNOT_DISCONNECT
0x2C00	COMMAND_SEQUENCE_ERROR	0x2C01	TOO_MANY_WINDOWS_SPECIFIED
0x2C02	INVALID_COMBINATION_OF_WINDOWS_SPECIFIED	0x2D00	OVERWRITE_ERROR_ON_UPDATE_IN_PLACE
0x2F00	COMMANDS_CLEARED_BY_ANOTHER_INITIATOR	0x3000	INCOMPATIBLE_MEDIUM_INSTALLED
0x3001	CANNOT_READ_MEDIUM_UNKNOWN_FORMAT	0x3002	CANNOT_READ_MEDIUM_INCOMPATIBLE_FORMAT
0x3003	CLEANING_CARTRIDGE_INSTALLED	0x3100	MEDIUM_FORMAT_CORRUPTED
0x3101	FORMAT_COMMAND_FAILED	0x3200	NO_DEFECT_SPARE_LOCATION_AVAILABLE
0x3201	DEFECT_LIST_UPDATE_FAILURE	0x3300	TAPE_LENGTH_ERROR
0x3600	RIBBON_INK_OR_TONER_FAILURE	0x3700	ROUNDED_PARAMETER
0x3900	SAVING_PARAMETERS_NOT_SUPPORTED	0x3A00	MEDIUM_NOT_PRESENT
0x3B00	SEQUENTIAL_POSITIONING_ERROR	0x3B01	TAPE_POSITION_ERROR_AT_BEGINNING_OF_MEDIUM
0x3B02	TAPE_POSITION_ERROR_AT_END_OF_MEDIUM	0x3B03	TAPE_OR_ELECTRONIC_VERTICAL_FORMS_UNIT_NOT_READY
0x3B04	SLEW_FAILURE	0x3B05	PAPER_JAM
0x3B06	FAILED_TO_SENSE_TOP_OF_FORM	0x3B07	FAILED_TO_SENSE_BOTTOM_OF_FORM
0x3B08	REPOSITION_ERROR	0x3B09	READ_PAST_END_OF_MEDIUM
0x3B0A	READ_PAST_BEGINNING_OF_MEDIUM	0x3B0B	POSITION_PAST_END_OF_MEDIUM
0x3B0C	POSITION_PAST_BEGINNING_OF_MEDIUM	0x3B0D	MEDIUM_DESTINATION_ELEMENT_FULL
0x3B0E	MEDIUM_SOURCE_ELEMENT_EMPTY	0x3D00	INVALID_BITS_IN_IDENTIFY_MESSAGE
0x3E00	LOGICAL_UNIT_HAS_NOT_SELF_CONFIGURED_YET	0x3F00	TARGET_OPERATING_CONDITIONS_HAVE_CHANGED
0x3F01	MICROCODE_HAS_BEEN_CHANGED	0x3F02	CHANGED_OPERATING_DEFINITION
0x3F03	INQUIRY_DATA_HAS_CHANGED	0x4000	DIAGNOSTIC_FAILURE_ON_COMPONENT_NN
0x4000	RAM_FAILURE	0x4100	DATA_PATH_FAILURE
0x4200	POWER_ON_OR_SELF_TEST_FAILURE	0x4300	MESSAGE_ERROR
0x4400	INTERNAL_TARGET_FAILURE	0x4500	SELECT_OR_RESELECT_FAILURE
0x4600	UNSUCCESSFUL_SOFT_RESET	0x4700	SCSI_PARITY_ERROR
0x4800	INITIATOR_DETECTED_ERROR_MESSAGE_RECEIVED	0x4900	INVALID_MESSAGE_ERROR
0x4A00	COMMAND_PHASE_ERROR	0x4B00	DATA_PHASE_ERROR
0x4C00	LOGICAL_UNIT_FAILED_SELF_CONFIGURATION	0x4E00	OVERLAPPED_COMMANDS_ATTEMPTED
0x5000	WRITE_APPEND_ERROR	0x5001	WRITE_APPEND_POSITION_ERROR
0x5002	POSITION_ERROR_RELATED_TO_TIMING	0x5100	ERASE_FAILURE
0x5200	CARTRIDGE_FAULT	0x5300	MEDIA_LOAD_OR_EJECT_FAILED
0x5301	UNLOAD_TAPE_FAILURE	0x5302	MEDIUM_REMOVAL_PREVENTED
0x5400	SCSI_TO_HOST_SYSTEM_INTERFACE_FAILURE	0x5500	SYSTEM_RESOURCE_FAILURE
0x5700	UNABLE_TO_RECOVER_TABLE_OF_CONTENTS	0x5800	GENERATION_DOES_NOT_EXIST
0x5900	UPDATED_BLOCK_READ	0x5A00	OPERATOR_REQUEST_OR_STATE_CHANGE_INPUT
0x5A01	OPERATOR_MEDIUM_REMOVAL_REQUEST	0x5A02	OPERATOR_SELECTED_WRITE_PROTECT
0x5A03	OPERATOR_SELECTED_WRITE_PERMIT	0x5B00	LOG_EXCEPTION
0x5B01	THRESHOLD_CONDITION_MET	0x5B02	LOG_COUNTER_AT_MAXIMUM
0x5B03	LOG_LIST_CODES_EXHAUSTED	0x5C00	RPL_STATUS_CHANGE

SCSI Error Codes – Numerical Order			
0x5C01	SPINDLES_SYNCHRONIZED	0x5C02	SPINDLES_NOT_SYNCHRONIZED
0x6000	LAMP_FAILURE	0x6100	VIDEO_ACQUISITION_ERROR
0x6101	UNABLE_TO_ACQUIRE_VIDEO	0x6102	OUT_OF_FOCUS
0x6200	SCAN_HEAD_POSITIONING_ERROR	0x6300	END_OF_USER_AREA_ENCOUNTERED_ON_THIS_TRACK
0x6400	ILLEGAL_MODE_FOR_THIS_TRACK	0xA981	END_OF_MEDIUM
0xA982	INCORRECT_LENGTH_INDICATOR	0xA9DD	ERROR_RESOURCE_FAILURE
0xA9FF	UNKNOWN_ERROR	0xAA02	SRB_STATUS_ABORTED
0xAA03	SRB_STATUS_ABORT_FAIL	0xAA04	SRB_STATUS_ERROR
0xAA80	SRB_STATUS_INVALID_COMMAND	0xAA81	SRB_STATUS_NO_ADAPTER
0xAA82	SRB_STATUS_NO_DEVICE	0xAAE4	SRB_STATUS_FAILED_INIT
0xAAE5	SRB_STATUS_ASPI_IS_BUSY	0xAAE6	SRB_STATUS_BUFFER_TO_BIG
0xAB09	SRB_HASTAT_TIMEOUT	0xAB0B	SRB_HASTAT_COMMAND_TIMEOUT
0xAB0D	SRB_HASTAT_MESSAGE_REJECT	0xAB0E	SRB_HASTAT_BUS_RESET
0xAB0F	SRB_HASTAT_PARITY_ERROR	0xAB10	SRB_HASTAT_REQUEST_SENSE_FAILED
0xAB11	SRB_HASTAT_SELECTION_TIMEOUT	0xAB12	SRB_HASTAT_DATAOVERRUN_DATAUNDERRUN
0xAB13	SRB_HASTAT_UNEXPECTED_BUS_FREE	0xAB14	SRB_HASTAT_PHASE_ERROR
0xAC02	SRB_TARGSTAT_CHECK_CONDITION	0xAC04	SRB_TARGSTAT_CONDITION_MET
0xAC08	SRB_TARGSTAT_BUSY	0xAC22	SRB_TARGSTAT_COMMAND_TERMINATED
0xAC28	SRB_TARGSTAT_QUEUE_FULL	0xAD00	SENSE_NO_SENSE
0xAD01	SENSE_RECOVERED_ERROR	0xAD02	SENSE_NOT_READY
0xAD03	SENSE_MEDIUM_ERROR	0xAD04	SENSE_HARDWARE_ERROR
0xAD05	SENSE_ILLEGAL_REQUEST	0xAD06	SENSE_UNIT_ATTENTION
0xAD07	SENSE_DATA_PROTECT	0xAD08	SENSE_BLANK_CHECK
0xAD09	SENSE_VENDOR_SPECIFIC	0xAD0A	SENSE_COPY_ABORTED
0xAD0B	SENSE_ABORTED_COMMAND	0xAD0C	SENSE_EQUAL
0xAD0D	SENSE_VOLUME_OVERFLOW	0xAD0E	SENSE_MISCOMPARE
0xAD0F	SENSE_RESERVED		